class treeNode(object):

def \_\_init\_\_(self, value):

self.value = value

self.l = None

self.r = None

self.h = 1

class AVLTree(object):

def insert(self, root, key):

if not root:

return treeNode(key)

elif key < root.value:

root.l = self.insert(root.l, key)

else:

root.r = self.insert(root.r, key)

root.h = 1 + max(self.getHeight(root.l),

self.getHeight(root.r))

b = self.getBal(root)

if b > 1 and key < root.l.value:

return self.rRotate(root)

if b < -1 and key > root.r.value:

return self.lRotate(root)

if b > 1 and key > root.l.value:

root.l = self.lRotate(root.l)

return self.rRotate(root)

if b < -1 and key < root.r.value:

root.r = self.rRotate(root.r)

return self.lRotate(root)

return root

def lRotate(self, z):

y = z.r

T2 = y.l

y.l = z

z.r = T2

z.h = 1 + max(self.getHeight(z.l),

self.getHeight(z.r))

y.h = 1 + max(self.getHeight(y.l),

self.getHeight(y.r))

return y

def rRotate(self, z):

y = z.l

T3 = y.r

y.r = z

z.l = T3

z.h = 1 + max(self.getHeight(z.l),

self.getHeight(z.r))

y.h = 1 + max(self.getHeight(y.l),

self.getHeight(y.r))

return y

def getHeight(self, root):

if not root:

return 0

return root.h

def getBal(self, root):

if not root:

return 0

return self.getHeight(root.l) - self.getHeight(root.r)

def preOrder(self, root):

if not root:

return

print("{0} ".format(root.value), end="")

self.preOrder(root.l)

self.preOrder(root.r)

Tree = AVLTree()

root = None

root = Tree.insert(root, 1)

root = Tree.insert(root, 2)

root = Tree.insert(root, 3)

root = Tree.insert(root, 4)

root = Tree.insert(root, 5)

root = Tree.insert(root, 6)

# Preorder Traversal

print("Preorder traversal of the",

"constructed AVL tree is")

Tree.preOrder(root)

print()